ELSEVIER

Discrete Applied Mathematics 87 (1998) 25–32

DISCRETE APPLIED MATHEMATICS

The cyclic cutwidth of trees

J.D. Chavez*,[1], R. Trapp[1]

Department of Mathematics, California State University, San Bernardino, 5500 University Parkway, San Bernardino, CA 92407, USA

Received 24 June 1997; received in revised form 29 December 1997; accepted 2 February 1998

Abstract

We consider the problem of embedding trees into cyclic host graphs in such a way as to minimize the cutwidth, the maximum number of edges along any point in the cycle. We show that the cyclic cutwidth of trees equals the linear cutwidth.

Keywords: Graph labeling problems; Trees

1. Introduction

In recent years, increasing interest has been shown in a variety of graph labeling problems. Given a graph, one may wish to label, or linearly arrange, the vertices in order to optimize some parameter, such as the bandwidth, edge sum, or cutwidth. A good survey of early results may be found in [2]. One may think of a labeling of the vertices of a graph, G, as an embedding of G into a linear host graph, H. It was suggested in [2] that other host graphs, such as grids and cycles, should be considered as well. In [1], the problem of minimizing the cutwidth of the n-dimensional cube when the host graph is a rectangular grid is solved. Results concerning the bandwidth of various families of graphs when the host graph is a cycle are given in [4]. The main result of this paper involves the cutwidth of trees embedded into cyclic hosts. We show that the cutwidth for a tree embedded into a cycle is the same as that for the tree embedded into a linear host.

To define the cutwidth of G in H, we first define an H-layout of G.

Definition. An *H-layout* of G is an ordered pair (π, P_π) consisting of:

(i) A one-to-one correspondence π between the vertices of G and those of H, and

* Corresponding author. E-mail: jchavez@wiley.csusb.edu.

[1] Partially supported by CSUSB Professional Development Grant.

0166-218X/98/$19.00
PII S0166-218X(98)00098-5

(ii) A collection P_π of paths in H, one path joining $\pi(v)$ to $\pi(w)$ for each pair of adjacent vertices v and w in G.

The cutwidth $c_\pi(G)$ of the layout (π, P_π) is the maximum number of times an edge e of H appears in the set of paths P_π. Note that different path choices can exist for the same one-to-one correspondence π, and the corresponding cutwidths may be different. The cutwidth $c(G)$ of G in H is the minimum of the cutwidths c_π taken over all layouts (π, P_π) of G in H.

We will consider two host graphs, a linear chassis and a cycle. We denote the cutwidths of G in the respective hosts by $lcw(G)$ and $ccw(G)$, and will refer to them as the linear and cyclic cutwidths of G, respectively. Note that for these host graphs the choice for paths joining $\pi(v)$ to $\pi(w)$ is rather limited. For the linear case there is a unique choice that makes sense if one is trying to minimize cutwidth. If the host graph is a cycle, note that $\pi(v)$ and $\pi(w)$ cut the cycle into two paths. Hence there are two reasonable choices in this case. As any linear chassis can be embedded in a cycle, we have $lcw(G) \geqslant ccw(G)$. Moreover, it seems intuitively clear that strict inequality can occur when cycles in G which lay flat in the linear chassis are allowed to wrap around the cyclic chassis. In the absence of cycles, then, one might expect the linear and cyclic cutwidths to coincide. The purpose of this paper is to show that this is the case.

We now restrict our attention to the case where $G = T$ is a tree, and wish to prove that $lcw(T) = ccw(T)$. Many results on the linear cutwidth of trees have been given. In particular, the cutwidth of the complete k-level t-ary tree $T_{t,k}$ has been determined [5], and an $O(n \log n)$ algorithm for determining the cutwidth of an arbitrary tree is given in [7]. Thus the main theorem of this paper facilitates the use of known results in a new context.

Since $lcw(G) \geqslant ccw(G)$ for arbitrary graphs, it remains to show that $lcw(T) \leqslant ccw(T)$ in the case of trees. In order to do so we describe an algorithm which produces a linear layout of T from a cyclic one without increasing the cutwidth. The algorithm is described in Section 2, and the proof that it does not increase the cutwidth is given in Section 3.

2. The algorithm

In this section we describe a method for producing linear layouts of trees from cyclic ones. Let H_C denote the cyclic host graph, and label the vertices of H_C with the integers $1, 2, \ldots, n$ in counterclockwise fashion. We denote the edges of H_C by $e_1, \ldots, e_n$, where e_i is incident to the vertices labeled $i-1$ and i for $i \neq 1$, and e_1 is incident to vertices labeled 1 and n. We consider the edges of H_C to be directed, all oriented in the counterclockwise direction.

The algorithm uses a cyclic layout (π_C, P_{π_C}) to associate ordered pairs of integers with the vertices of T. Using π_C, one obtains a labeling π of T by composing π_C with

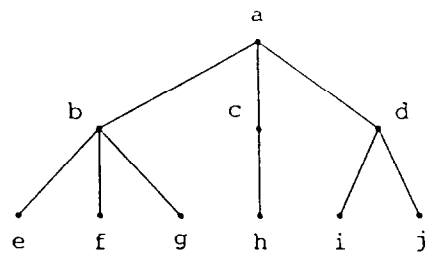

Fig. 1. The tree T.

the labeling of the vertices of H_C. Thus $\pi(v)$ is the label of the vertex $\pi_C(v)$ in H_C. For each vertex, v, of T, the second coordinate of the ordered pair associated with v is $\pi(v)$. In other words, the second coordinate associated with a vertex v of T is the position of its image under π_C in the counterclockwise ordering of the cycle H_C. We designate by r the vertex of T such that $\pi(r)=1$, and think of r as a root of T.

Before obtaining the first coordinate of the ordered pair, we define the image in H_C of a path in T. Recall that there is a path p_e in P_{π_C} for each edge e of T. The *image* in H_C of a path p_T in T is the path obtained by concatenating the paths p_e for each edge e in p_T. Now let v be a vertex of T, and let $\vec{p}_v$ be the unique path in T from r to v. The first coordinate of v comes from the image of $\vec{p}_v$ in H_C. It counts the number of times that e_1 is traversed by the image of $\vec{p}_v$ in H_C in the direction of its counterclockwise orientation, minus the number of times that e_1 is traversed in the clockwise direction. Thus, the first coordinate of the ordered pair associated with v in T is the net number of times the image in H_C of the unique path from r to v crosses the edge e_1. We call the first coordinate the *wrapping number* since it can be thought of as counting the net number of times the image of $\vec{p}_v$ wraps around the cycle H_C.

To obtain a linear layout for vertices of T simply use the lexicographic ordering on the associated ordered pairs.

Example. Consider the graph T in Fig. 1, which is embedded in the cycle shown in Fig. 2.

Note that vertex c of T is designated as the root in our algorithm, and e_1 is the edge connecting c to e in the host graph, H_C. The first coordinate of vertex e, for example, is -2 since the unique path from c to e passes through edge e_1 of the host graph twice in the clockwise direction.

We use the lexicographic order on the associated pairs shown in Fig. 2 to construct the embedding into the linear host H_L as shown in Fig. 3. Observe that the linear cutwidth of the embedding induced by the algorithm is less than the cyclic cutwidth.

In [4] a similar algorithm is constructed that embeds the vertices of a graph into the integers. While the linear orders induced by the two algorithms are the same, our proof that linear cutwidth equals cyclic cutwidth for trees is quite different from their proof about the bandwidth of trees.

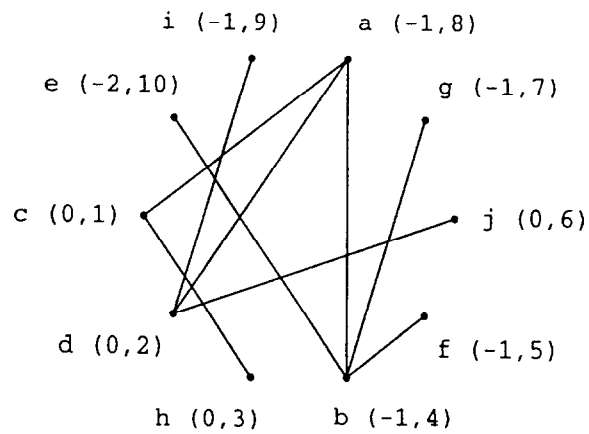

Fig. 2. T embedded in the cyclic host, H_C.

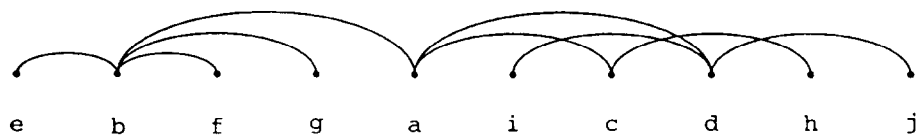

Fig. 3. T embedded in the linear host, H_L.

3. The cyclic cutwidth for trees

Before proving that the above algorithm does not increase the cutwidth, some observations are in order. In particular, we are interested in how an edge e of T can be placed in H_C if we know the ordered pairs associated to its endpoints. Note that two vertices of H_C cut the cycle into two paths. One path contains the edge e_1 and the other does not. We have the following lemma:

Lemma 1. *Let v,w be adjacent vertices in T with associated ordered pairs $(a,b),(c,d)$ where $(a,b)<(c,d)$. If e is the edge incident to v and w, then we have the following:*
(i) *If $a<c$, then the image of e in H_C contains e_1.*
(ii) *If $a=c$, then e_1 is not contained in the image of e in H_C.*

Proof. First suppose that $a<c$. Since the wrapping numbers differ the path representing e in H_C must contain e_1. If $a=c$, then the wrapping numbers are the same. Thus the path representing e in H_C does not contain e_1. □

Remark. A consequence of this lemma is that if $a<c$, then $c=a+1$. This follows since the path crosses e_1 exactly once. Another way to say this is that the wrapping numbers of adjacent vertices in T can differ by at most one.

The above lemma and remark will be useful in the proof of the following theorem.

Theorem 1. *If T is a tree then $lcw(T) = ccw(T)$.*

Proof. Recall that for arbitrary graphs the linear cutwidth is an upper bound for the cyclic cutwidth, thus it remains to show that $ccw(T) \geqslant lcw(T)$. In order to prove this, we show that the above algorithm produces a linear layout that does not increase the cutwidth.

Let H_L denote the linear host graph, and let (π_L, P_{π_L}) be the linear layout of T induced by (π_C, P_{π_C}) using the algorithm of Section 2. Recall that $c_{\pi_L}(T)$ is the maximum number of occurrences of paths in P_{π_L} along any single edge in H_L, and that each path in P_{π_L} is the image of an edge in T. Let e_L be an edge of H_L which occurs in a maximum number of paths in P_{π_L}, and let s and t be the vertices of T whose images $\pi_L(s)$ and $\pi_L(t)$ are the endpoints of e_L.

As in the remarks preceding Lemma 1, the images $\pi_C(s)$ and $\pi_C(t)$ of s and t in H_C cut the cycle into two disjoint paths. We denote the path containing e_1 by p_1 and the other path by p_2. We will show that each edge e of T whose image in H_L contains e_L must also yield an image in H_C which contains a fixed path. The fixed path is either p_1 or p_2, but not both, depending on the ordered pairs (f,g) and (h,k) associated with s and t. In either case, the cutwidth along the fixed path in H_C will then be at least as large as $c_{\pi_L}(T)$, and we will have shown that $c_{\pi_L}(T) \leqslant c_{\pi_C}(T)$; thereby proving the theorem.

We assume that $(f,g) < (h,k)$, or that $\pi_L(s)$ is the left-hand endpoint of e_L and $\pi_L(t)$ is the right. Now let e be any edge of T whose image in H_L contains e_L, and let v and w be the endpoints of e in T. Let (a,b) and (c,d) be the ordered pairs associated with v and w, respectively, and assume that $(a,b) < (c,d)$. Then in the lexicographic ordering we have $(a,b) \leqslant (f,g) < (h,k) \leqslant (c,d)$, and there are two cases to consider.

If $f < h$ we wish to show that the image of e in H_C contains the path p_1. Since wrapping numbers of adjacent vertices differ by at most one, and since we are assuming that $f < h$ and $(a,b) \leqslant (f,g) < (h,k) \leqslant (c,d)$, we have that $a = f$, $b \leqslant g$, $h = c$ and $k \leqslant d$ (see Fig. 4). Thinking of e as oriented from v to w, one sees that the image of e in H_C starts at $\pi_C(v)$, moves counterclockwise past $\pi_C(s)$ (since $b \leqslant g$) and the edge e_1 (since $f < h$), then past $\pi_C(t)$ (since $k \leqslant d$) to the vertex $\pi_C(w)$. Hence, it contains the entire path p_1, the path between $\pi_C(s)$ and $\pi_C(t)$ which contains the edge e_1.

We conclude that if $f < h$, then any edge e whose image in H_L contains e_L yields an image in H_C which contains all of p_1. Thus the cutwidth along all of p_1 in H_C is at least as large as $c_{\pi_L}(T)$, and we have $c_{\pi_L}(T) \leqslant c_{\pi_C}(T)$ in this case.

If $f = h$, we wish to show that the image in H_C of e contains all of p_2 (recall that the edge e is one whose image in H_L is a path containing e_L). Again we must have that $(a,b) \leqslant (f,g) < (h,k) \leqslant (c,d)$, and since wrapping numbers of adjacent vertices differ by at most one either $a = c$ or $a = c - 1$. If $a = c$, we must have equality of all wrapping numbers. If a and c differ, we could have $a < f$ or $a = f$. In summary, we

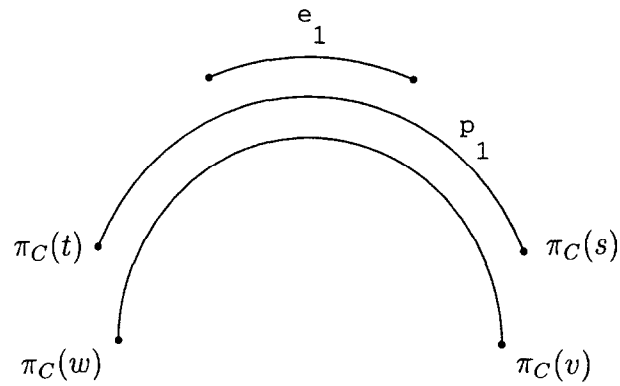

Fig. 4.

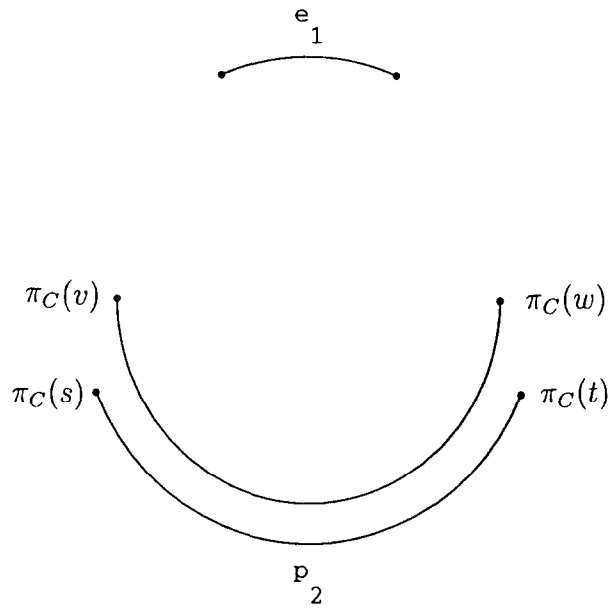

Fig. 5. $a=f=h=c$.

have the three cases $a=f=h=c$, $a=f-1<f=h=c$, and $a=f=h<h+1=c$ (see Figs. 5–7).

First, if $a=f=h=c$, then Lemma 1 implies that e yields the path in H_C joining $\pi_C(v)$ and $\pi_C(w)$ which does not contain e_1. But the second coordinate tells the ordering on the cycle and we must have $b\leqslant g<k\leqslant d$, hence the image of e in H_C is a path which contains all of p_2. Now suppose that $a<f=h=c$, which implies that $g<k\leqslant d$. In this case the image in H_C of e must cross e_1 in a counterclockwise fashion when oriented from $\pi_C(v)$ to $\pi_C(w)$. After crossing e_1 it must travel past $\pi_C(s)$ and $\pi_C(t)$ in order to reach $\pi_C(w)$. Hence it contains all of p_2. The case where $a=f=h<c$

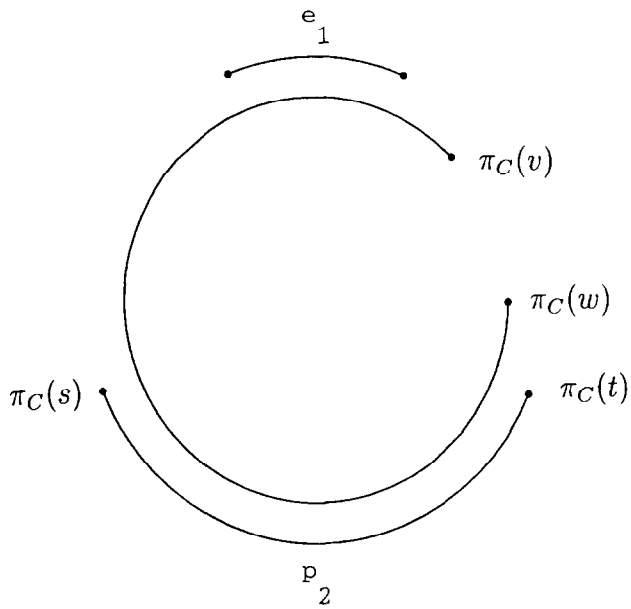

Fig. 6. $a = f - 1 < f = h = c$.

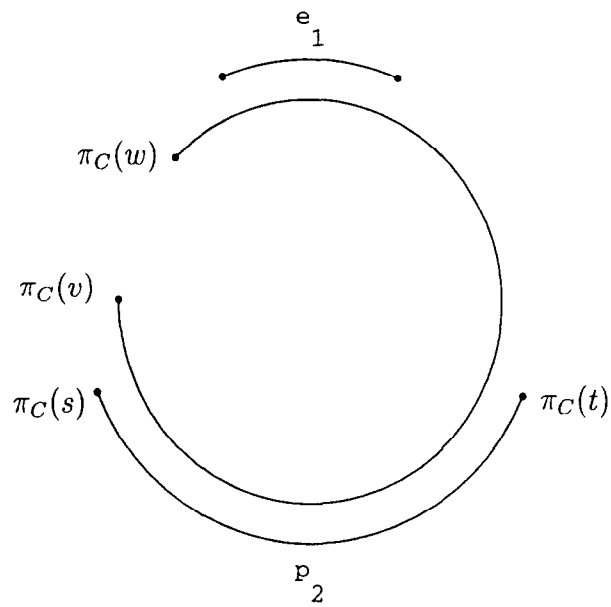

Fig. 7. $a = f = h < h + 1 = c$.

is similar. Thus, we have that the image of e in H_C must contain all of p_2 if $f = h$, proving that $c_{\pi_L}(T) \leqslant c_{\pi_C}(T)$ in this case as well.

References

[1] S. Bezrukov, J.D. Chavez, L.H. Harper, M. Röttger, U-P Schroeder, The congestion of n-cube layout on a rectangular grid, Discrete Math., in press.

[2] F.R.K. Chung, Labelings of Graphs, in: Selected Topics in Graph Theory, vol. 3, Academic Press, New York, 1988, pp. 151–168.

[3] L.H. Harper, Optimal assignments of numbers to vertices, J. Soc. Ind. Appl. Math. 12 (1964) 131–135.

[4] J. Hromkovič, V. Müller, O. Sýkora, I. Vrťo, On embedding interconnection networks into rings of processors, in: Lecture Notes in Computer Science, vol. 605, Springer, Berlin, 1992, pp. 53–62.

[5] T. Lengauer, Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees, SIAM J. Algebra Discrete Meth. 3 (1982) 99–113.

[6] K. Nakano, Linear layouts of generalized hypercubes, in: Lecture Notes in Computer Science, vol. 790, Springer, Berlin, 1994, pp. 364–371.

[7] M. Yannakakis, A polynomial algorithm for the min-cut linear arrangement of trees, in: Proceedings of the 24th IEEE Symposium on Foundation of Computer Science, 1983, pp. 274–281.